IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appellant: Stephen L. Bade et al.

Assignee: Synopsys, Inc.

Title: Method and System for Virtual Prototyping

Serial No.: 09/872,435 File Date: June 1, 2001

Examiner: Cuong V. Luu Art Unit: 2128

Docket No.: SYN-0472

March 12, 2007

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

This Appeal Brief is in support of the Notice of Appeal dated January 22, 2007.

/

/

/

/

/

/

/

/

/

/

/

INDEX

I. REAL PARTY IN INTEREST

The real party in interest is the assignee, Synopsys, Inc., pursuant to the Assignment recorded in the U.S. Patent and Trademark Office on October 15, 2001 on Reel 012269, Frame 0304 and on October 24, 2006 on Reel 018430, Frame 0918.

II. RELATED APPEALS AND INTERFERENCES

Based on information and belief, there are no other appeals or interferences that could directly affect or be directly affected by or have a bearing on the decision by the Board of Patent Appeals in the pending appeal.

III. STATUS OF CLAIMS

Claims 37-78 and 89 are pending. Claims 37-78 and 89 stand rejected.

In the present paper, rejected Claims 37-72, 74-78, and 89 are appealed. Note that Claim 73 is objected to and therefore is not appealed.

Pending Claims 1-89 are listed in Appendix XIII. Note that Claims 1-36, 79-88 are withdrawn, but listed for completeness.

IV. STATUS OF AMENDMENTS

All claim amendments were entered.

V. SUMMARY OF CLAIMED SUBJECT MATTER

A concise explanation of the subject matter defined in each of the independent claims involved in the appeal (i.e. **Claims 37, 52, 64, 68, and 74**) is provided below. This concise explanation refers to the specification by paragraph, page, and line numbers, as well as to the drawings, if any, by reference numbers/characters. Appellant notes that all citations to the Specification are based on the application as originally filed.

As described by in the **Specification, paragraph 0104, page 36, line 15 to page 37, line 17:**

The integrated design environment (IDE) of the present invention permits a model of an embedded system to be created in a comparatively short time, e.g. within a few hours to a few days for many typical embedded systems. The short model development time is due, in part, to the GUI, which can include numerous features and a powerful finite state machine (FSM) design language for rapidly creating FSM models and forming re-usable models. However, another factor is that the level of abstraction is high enough to not require low-level hardware implementation details while still having sufficient software-centric details to execute a user application. This makes the IDE consistent with developing a virtual prototype that is a useful functional representation of an embedded system within the comparatively small time allotted in typical projects for evaluating IP blocks and creating an architecture definition. The resulting virtual prototype can be used as a development target for embedded software early in the development cycle, e.g. before any hardware or prototype is available. Advantageously, a user can load production quality code (e.g. binary code compiled for a specific processor) for execution on

the virtual embedded system. The high level of abstraction facilitates sharing models of embedded systems with others, while preserving trade secrets.

Claim 37. In a computer system having a graphical interface (GUI) and a design language for forming a finite state machine (FSM) representation of a hardware partition of an embedded system, a method of designing an embedded system, the method comprising:

forming a library of processor cores **[Fig. 9, 902; Specification, paragraph 0094, page 28, lines 5-21]** including an instruction set accurate simulator for each of the processor cores in the library **[Specification, paragraph 0084, page 22, lines 9-12, paragraph 0097, page 30, line 13 to page 31, line 15]**;

responsive to a first sequence of user commands, selecting at least one of the processor cores from the library as a target processor core **[Specification, paragraph 0143, page 59, lines 11-16; Fig. 6, 656, 636; Specification, paragraph 0202, page 89, line 11 to page 90, line 3; Figs. 52-53; Specification, paragraph 0213, page 99, lines 4-17, paragraph 0214, page 99, line 18 to page 100, line 13]**;

responsive to a second sequence of user commands, forming a virtual embedded system **[Fig. 6, 658, 642; Specification, paragraph 0202, page 89, line 11 to page 90, line 3; Figs. 52-53; Specification, paragraph 0214, page 99, line 18 to page 100, line 13]** including an instruction set accurate simulator of a target processor core and coupling read, write, and interrupt signals of the instruction set accurate simulator with an FSM simulation of at least one hardware element **[Fig. 37; Specification, paragraph 0103, page 35, line 18 to page 36, line 14, paragraph 0145, page 60, lines 3-7, paragraph 0147, page 60,**

lines 12-16, paragraph 0149, page 61, lines 6-9, paragraph 0152, page 61, line 15 to page 62, line 11], wherein generating said FSM simulation comprises applying a design language having at least one graphical symbol and adapted to form a finite state machine representation of electronic hardware, each graphical symbol of the design language having a graphical portion and a user-definable textual portion defining the behavior of the graphical symbol **[Figs. 19A, 19B; Specification, paragraph 0099, page 32, line 11 to page 33, line 7, paragraph 0116, page 44, line 6 to page 45, line 9]**;

responsive to a request from the user, loading an executable binary file of a software application compiled for the target processor **[Specification, paragraph 0140, page 58, lines 2-10; Fig. 6, 660; Specification, paragraph 0202, page 89, line 11 to page 90, line 3; Fig. 54, 5404; Specification, paragraph 0214, page 99, line 18 to page 100, line 13]**;

executing a simulation of the virtual embedded system running the software application **[Specification, paragraph 0140, page 58, lines 2-10; Fig. 6, 662; Specification, paragraph 0202, page 89, line 11 to page 90, line 3; Figs. 54, 55; Specification, paragraph 0214, page 99, line 18 to page 100, line 13]**; and

responsive to a user request, displaying on the GUI a graphical representation of the execution of the software application on the virtual embedded system **[Specification, paragraph 0181, page 76, lines 12-17, paragraph 0182, page 76, line 19 to page 77, line 2, paragraph 0183, page 77, lines 3-7]** that includes a software debugger interface to debug the loaded software and a virtual test-bench associated with the GUI and adapted to interact with the simulation **[Fig. 4A, 442; Specification, paragraph 0085, page 23, lines 1-8; Fig. 14; Specification, paragraph 0157, page 64, line 13 to page 65, line**

5; Specification, paragraph 0169, page 71, lines 4-10, paragraph 0171, page 71, line 16 to page 72, line 3; Figs. 34A-34C; paragraph 0175, page 73, lines 6-21; Specification, paragraph 0176, page 74, lines 1-16; Figs 36A-36C; specification, Fig. 46, paragraph 0179, page 75, lines 6-20; Fig. 55, 5510, 5520, 5530; Specification, paragraph 0214, page 99, line 18 to page 100, line 13], wherein the virtual test-bench is created using a test-bench builder **[Fig. 9, 910; Specification, paragraph 0094, page 28, lines 5-21]** for generating a graphical representation of at least one interactive test-bench and for selecting signals or variables associated with the FSM and to be coupled to a graphical representation of a user interface for each interactive test-bench, to emulate user input to and device output from the virtual embedded system **[Specification, paragraph 0101, page 34, lines 4-20; Figs. 34A-34C; Specification, paragraph 0175, page 73, lines 6-21; Specification, paragraph 0176, page 74, lines 1-16]**.

Claim 52. A computer implemented method of embedded system design **[FIG. 4A; Specification, paragraph 0082, page 21, lines 5-11]**, the method comprising:

selecting an instruction set accurate simulator of a target processor core **[Specification, paragraph 0084, page 22, lines 9-12, paragraph 0097, page 30, line 13 to page 31, line 15, paragraph 0143, page 59, lines 11-16; Figs. 52-53; Specification, paragraph 0213, page 99, lines 4-17, paragraph 0214, page 99, line 18 to page 100, line 13]**;

generating a virtual hardware component that is a finite state machine (FSM) representation of at least one hardware component **[Fig. 13, paragraph 98, page 31, line 16 to page 99, line 10, paragraph 0099, page 32, line 11 to page 33, line 7]**, said generating comprising applying a design language having at

least one graphical symbol and adapted to form an FSM representation of electronic hardware, each graphical symbol of the design language having a graphical portion and a user-definable textual portion defining the behavior of the graphical symbol **[Figs. 19A, 19B; Specification, paragraph 0099, page 32, line 11 to page 33, line 7, paragraph 0116, page 44, line 6 to page 45, line 9]**;

linking read, write, and interrupt signals of the instruction set accurate simulator of the target processor core with corresponding signals of the at least one virtual hardware component to form a virtual embedded system **[Fig. 37; Specification, paragraph 103, page 35, line 18 to page 36, line 14, paragraph 0145, page 60, lines 3-7, paragraph 0147, page 60, lines 12-16, paragraph 0149, page 61, lines 6-9, paragraph 0152, page 61, line 15 to page 62, line 11]**;

creating a virtual test bench using a test bench builder for generating a graphical representation of at least one interactive test bench and for selecting signals or variables to be coupled to a graphical representation of a user interface for each interactive test bench **[Fig. 9, 910; Specification, paragraph 0094, page 28, lines 5-21; Fig. 14; Specification, paragraph 0157, page 64, line 13 to page 65, line 5; Figs. 34A-34C; Specification, paragraph 0175, page 73, lines 6-21; Specification, paragraph 0176, page 74, lines 1-16]**;

coupling the virtual test bench to at least one signal or variable of the virtual embedded system to simulate a human/machine interface **[Figs. 34A-34C; Specification, paragraph 0175, page 73, lines 6-21; Specification, paragraph 0176, page 74, lines 1-16]**; and

coupling a software debugger to the virtual embedded system that is configured to load and run on the virtual embedded system at least one binary program executable of a software

application compiled for the target processor core **[Specification, paragraph 0019, page 10, line 14 to page 12, line 2; Fig. 4A, 442; Specification, paragraph 0085, page 23, lines 1-8; Fig. 53, 5520; Specification, paragraph 0214, page 99, line 18 to page 100, line 13]**.

Claim 64. A method of designing an embedded system, the method comprising:

defining a system architecture of the embedded system **[Specification, paragraph 0139, page 57, lines 4-18]**;

generating a finite state machine (FSM) representation of at least one hardware element **[Fig. 13, paragraph 0098, page 31, line 16 to page 99, line 10, paragraph 0099, page 32, line 11 to page 33, line 7]**, said generating comprising applying a design language having at least one graphical symbol and adapted to form a finite state machine representation of electronic hardware, each graphical symbol of the design language having a graphical portion and a user-definable textual portion defining the behavior of the graphical symbol **[Figs. 19A, 19B; Specification, paragraph 0099, page 32, line 11 to page 33, line 7, paragraph 0116, page 44, line 6 to page 45, line 9]**;

designing a virtual prototype of the embedded system having an instruction set accurate simulator of a processor core **[Specification, paragraph 0084, page 22, lines 9-12, paragraph 0097, page 30, line 13 to page 31, line 15, paragraph 0143, page 59, lines 11-16; Figs. 52-53; Specification, paragraph 0213, page 99, lines 4-17, paragraph 0214, page 99, line 18 to page 100, line 13]** and coupling read, write, and interrupt signals of the instruction set accurate simulator with said FSM representation of at least one hardware element **[Fig. 37; Specification, paragraph 0103, page 35, line 18 to page 36, line 14]**;

creating a virtual test bench having a graphical representation of a human/machine interface **[Figs. 34A-34C; Specification, paragraph 0175, page 73, lines 6-21; Specification, paragraph 0176, page 74, lines 1-16]** for interacting with the embedded system using a test bench builder for generating a graphical representation of at least one interactive test bench and for selecting signals or variables associated with the FSM and to be coupled to a graphical representation of a user interface for each interactive test bench **[Fig. 9, 910; Specification, paragraph 0094, page 28, lines 5-21; Fig. 14; Specification, paragraph 0157, page 64, line 13 to page 65, line 5; Figs. 34A-34C; Specification, paragraph 0175, page 73, lines 6-21; Specification, paragraph 0176, page 74, lines 1-16]**;

coupling the virtual prototype to a software debugger having a debugging interface and to the virtual test bench **[Specification, paragraph 0019, page 10, line 14 to page 12, line 2]**;

developing at least one software application for the processor core **[Specification, paragraph 0085, page 23, lines 1-8]**;

loading binary program code compiled from the at least one software application for execution on the virtual prototype **[Specification, paragraph 0140, page 58, lines 2-10; Fig. 6, 660; Specification, paragraph 0202, page 89, line 11 to page 90, line 3; Fig. 54, 5405; Specification, paragraph 0214, page 99, line 18 to page 100, line 13]**; and

initiating a simulation of the virtual prototype executing the at least one software application **[Specification, paragraph 0140, page 58, lines 2-10; Fig. 6, 662; Specification, paragraph 0202, page 89, line 11 to page 90, line 3; Figs. 54, 55; Specification, paragraph 0214, page 99, line 18 to page 100,**

line 13].

Claim 68. A method of providing information to potential suppliers for procuring a good or service associated with an embedded system, the method comprising:

defining a system architecture of the embedded system **[Specification, paragraph 0139, page 57, lines 4-18]**;

designing a virtual prototype of the embedded system, the virtual prototype having an instruction set accurate simulator of a target processor core **[Specification, paragraph 0084, page 22, lines 9-12, paragraph 0097, page 30, line 13 to page 31, line 15, paragraph 0143, page 59, lines 11-16; Figs. 52-53; Specification, paragraph 0213, page 99, lines 4-17, paragraph 0214, page 99, line 18 to page 100, line 13]** and a finite state machine (FSM) representation of a hardware element within the embedded system **[Fig. 13, paragraph 98, page 31, line 16 to page 99, line 10, paragraph 0099, page 32, line 11 to page 33, line 7]**, the FSM representation configured to couple memory read/write requests and interrupt signals with the instruction set accurate simulator of the target processor core **[Fig. 37; Specification, paragraph 0103, page 35, line 18 to page 36, line 14]**, wherein generating said FSM representation comprises applying a design language having at least one graphical symbol and adapted to form a finite state machine representation of electronic hardware, each graphical symbol of the design language having a graphical portion and a user-definable textual portion defining the behavior of the graphical symbol **[Figs. 19A, 19B; Specification, paragraph 0099, page 32, line 11 to page 33, line 7, paragraph 0116, page 44, line 6 to page 45, line 9]**;

creating a virtual test bench having a graphical representation of a human/machine interface **[Figs. 34A-34C;**

Specification, paragraph 0175, page 73, lines 6-21; Specification, paragraph 0176, page 74, lines 1-16] for interacting with the embedded system using a test bench builder for generating a graphical representation of at least one interactive test bench and for selecting signals or variables associated with the FSM and to be coupled to a graphical representation of a user interface for each interactive test bench **[Fig. 9, 910; Specification, paragraph 0094, page 28, lines 5-21; Fig. 14; Specification, paragraph 0157, page 64, line 13 to page 65, line 5; Figs. 34A-34C; Specification, paragraph 0175, page 73, lines 6-21; Specification, paragraph 0176, page 74, lines 1-16]**;

coupling the virtual prototype to the virtual test bench for emulating user interaction with the embedded system **[Specification, paragraph 0019, page 10, line 14 to page 12, line 2]**;

publishing the virtual prototype as a functional specification from which a vendor may initiate a simulation of the operation of the embedded system **[Figs. 6, 7, 8; Specification, paragraph 0093, page 27, line 13, to page 28, line 4, paragraph 0201, page 88, line 8 to page 89, line 10]**.

Claim 74. A computer-implemented method for a vendor to acquire information for the procurement of a good or service related to an embedded system project, the method comprising:

accessing a database of virtual prototypes of embedded systems, each of the virtual prototypes having a processor simulator **[Fig. 9, 902; Specification, paragraph 0094, page 28, lines 5-21]**, a finite state machine (FSM) representation of hardware peripherals **[Fig. 13, paragraph 0098, page 31, line 16 to page 99, line 10, paragraph 0099, page 32, line 11 to page 33, line 7]**, and a virtual test bench emulating a human/machine

interface **[Fig. 9, 910; Figs. 34A-34C; Specification, paragraph 0175, page 73, lines 6-21; Specification, paragraph 0176, page 74, lines 1-16]** for interacting with a simulation of the operation of the virtual prototype,

wherein generating said FSM representation comprises applying a design language having at least one graphical symbol and adapted to form a finite state machine representation of electronic hardware **[Fig. 13, paragraph 98, page 31, line 16 to page 99, line 10, paragraph 0099, page 32, line 11 to page 33, line 7]**, each graphical symbol of the design language having a graphical portion and a user-definable textual portion defining the behavior of the graphical symbol **[Figs. 19A, 19B; Specification, paragraph 0099, page 32, line 11 to page 33, line 7, paragraph 0116, page 44, line 6 to page 45, line 9]**, and

wherein the virtual test bench is created using a test bench builder for generating a graphical representation of at least one interactive test bench and for selecting signals or variables associated with the FSM and to be coupled to a graphical representation of a user interface for the interactive test bench **[Fig. 9, 910; Specification, paragraph 0094, page 28, lines 5-21; Fig. 14; Specification, paragraph 0157, page 64, line 13 to page 65, line 5; Figs. 34A-34C; Specification, paragraph 0175, page 73, lines 6-21; Specification, paragraph 0176, page 74, lines 1-16]**;

instantiating an instance of one of the virtual prototypes **[Fig. 7; Specification, paragraph 0205, page 93, lines 2-5]**; and

evaluating the virtual prototype **[Specification, paragraph 0019, page 10, line 14 to page 12, line 2]**.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The following issues are presented to the Board of Appeals for decision:

A. Whether Claims 37-39 and 49-51 are patentable under 35 U.S.C. 103(a) over E.U. Application 96870126.8 (Rompaey), "Datasheet Interactive Simulation Library, 072597CM5.97 1997" (Cadence), and U.S. Patent 6,263,302 (Hellestrand).

B. Whether Claims 40-42 are patentable under 35 U.S.C. 103(a) over Rompaey, Cadence, Hellestrand, and U.S. Publication 2004/0250083 (Schwab).

C. Whether Claims 43-48 are patentable under 35 U.S.C. 103(a) over Rompaey, Cadence, Hellestrand, and U.S. Patent 6,094,654 (Van Huben).

D. Whether Claims 52-59, 64, and 89 are patentable under 35 U.S.C. 103(a) over Rompaey and Cadence.

E. Whether Claims 60-63 are patentable under 35 U.S.C. 103(a) over Rompaey, Cadence, and Van Huben.

F. Whether Claims 65-67 are patentable under 35 U.S.C. 103(a) over Rompaey, Cadence, and U.S. Publication 2005/0193280 (Schubert).

G. Whether Claims 68-70 are patentable under 35 U.S.C. 103(a) over Rompaey, Cadence, and Van Huben.

H. Whether Claims 71-72 are patentable under 35 U.S.C. 103(a) over Rompaey, Cadence, Van Huben, and Schwab.

I. Whether Claims 74 and 76-78 are patentable under 35 U.S.C. 103(a) over Van Huben, Rompaey, and Cadence.

J. Whether Claim 75 is patentable under 35 U.S.C. 103(a) over Van Huben, Rompaey, Cadence, and Schwab.

VII. ARGUMENTS

A. Claims 37-39 and 49-51 are patentable under 35 U.S.C. 103(a) over E.U. Application 96870126.8 (Rompaey), "Datasheet Interactive Simulation Library, 072597CM5.97 1997" (Cadence), and U.S. Patent 6,263,302 (Hellestrand).

1. Rompaey: Overview

Rompaey teaches a hardware/software co-design environment and design methodology based on a data-model that can specify, simulate, and synthesize heterogeneous hardware/software architectures from a heterogeneous specification. Col. 7, lines 31-35. In one method taught in col. 34, lines 9-54, Rompaey defines a set of primitive objects representing the specification of a digital system. This defining includes describing the specification of the digital system in processes (each process representing a functional aspect of the system), defining ports and channels that connects ports, and defining the communication semantics of the ports by a protocol. The

processes, ports, channels, and protocol are all primitive objects. After the primitive objects are defined, hierarchical objects are created, wherein the hierarchical objects are refinements of the primitive objects and have more detail, while preserve aspects of communication semantics. Rompaey allocates one or more hardware components, such as programmable processors and non-programmable processors, and assigns processes to the hardware components. The processes assigned to a programmable processor are a software sub-system, whereas all other processes are hardware sub-systems. Finally, Rompaey selects I/O models for the ports of the software sub-system (to connect ports to the interface of the programmable processor). At this point, Rompaney can simulate the system.

2. Cadence: Overview

Cadence teaches an interactive simulation library (ISL) that allows a user to create virtual instruments (e.g. oscilloscopes, spectrum analyzers, and signal generators, which can be used to test and evaluate system designs). The features and displays for an interactive control window can be specified by having the user choose ISL primitives and wire them together into a block diagram. This block diagram can be incorporated in a signal flow block diagram for a system under test. During simulation, the interactive control window automatically pops up and begins to monitor the performance of the system.

3. Hellestrand: Overview

In col. 6, lines 39-58, Hellestrand teaches a method of simulating on a host computer system the execution of a user program on a target processor having a cache. This method includes decomposing the user program into linear blocks and then determining the linear block timing information including

the time delays that would be incurred executing each linear block of the user program on the target processor with no cache misses. Those parts of the user program that include memory references that might require a cache lookup can be identified. Hooks can be inserted into the user program to invoke, at run time, a cache simulator that simulates the operation of the cache for the memory reference to account for cache misses in timing. The combined user program, linear block timing information, and inserted hooks can be executed on the host computer system. The execution on the host computer system can simulate the execution of the user program on the target processor and provide accurate execution timing that takes into account instruction timing and cache effects (e.g. pipeline effects for a processor that has a pipeline).

4. The limitations recited in Claims 37-39 and 49-51 are not taught by Rompaey, Cadence, and Hellestrand.

Claim 37 recites in part (emphasis added):

> responsive to a second sequence of user commands, forming a virtual embedded system including an instruction set accurate simulator of a target processor core and coupling read, write, and interrupt signals of the instruction set accurate simulator with an FSM simulation of at least one hardware element, wherein generating said FSM simulation comprises applying a design language having at least one graphical symbol and adapted to form a finite state machine representation of electronic hardware, each graphical symbol of the design language having a graphical portion and a user-definable textual portion defining the behavior of the graphical symbol; … and
>
> responsive to a user request, displaying on the GUI a graphical representation of the execution of the software application on the virtual embedded system that includes a software debugger interface to debug the loaded software and a virtual test-bench associated with the GUI and adapted to interact with the simulation, wherein the virtual test-bench is created using a test-bench builder for generating a

graphical representation of at least one interactive test-bench and for selecting signals or variables associated with the FSM and to be coupled to a graphical representation of a user interface for each interactive test-bench, to emulate user input to and device output from the virtual embedded system.

Appellant respectfully submits that the cited references fail to disclose or suggest these limitations. For example, Rompaey teaches a hardware/software co-design environment that allows specifying, simulating, and synthesizing heterogeneous hardware/software architectures from a heterogeneous specification. Col. 7, lines 31-35. This environment is based on encapsulating existing hardware and software compilers and allows for the interactive synthesis of hardware/software as well as hardware/hardware interfaces. Col. 7, lines 35-39.

The Final Office Action cites the following passages of Rompaey as teaching forming the virtual embedded system that includes generating a finite state machine (FSM) simulation: col. 9, lines 19-22; col. 20, line 55 to col. 21, line 14; and col. 11, lines 5-6. Appellant respectfully traverses this characterization. Col. 9, lines 19-22 teach that a hybrid simulation includes both hardware implementations and computer simulations, wherein the hardware implementation can include hardware and software subsystems (the software subsystems being executed on the hardware subsystems). This simulation does not teach an FSM, much less the recited generating of the FSM simulation. Col. 20, line 55 to col. 21, line 14 teach that the ports 75 of the software model 71 (see FIG. 7) for an ARM processor have primitive protocols, wherein the software model contains a behavioral description that allows compiling a software host language encapsulation into machine code. This software model does not teach an FSM, much less the recited generating of the FSM simulation. Col. 11, lines 5-6 teach that

FIG. 11 is a schematic representation of a pager application. FIG. 11 shows Remote Procedure Call (RPC) communication for the pager design. Col. 24, lines 50-51. In this RPC communication, blocks 91-101 correspond to processes 89 (i.e. tracking and acquisition, frame extraction, correlator and noise estimator, and user interface) shown in FIG. 10. Col. 24, lines 51-53. However, these blocks do not teach an FSM, much less the recited generating of the FSM simulation.

Thus, Appellant respectfully submits that these cited passages of Rompaey fail to disclose or suggest the recited step of forming the virtual embedded system including an instruction set accurate simulator of a target processor core and coupling read, write, and interrupt signals of the instruction set accurate simulator with an FSM simulation of at least one hardware element, wherein generating said FSM simulation comprises applying a design language having at least one graphical symbol and adapted to form a finite state machine representation of electronic hardware, each graphical symbol of the design language having a graphical portion and a user-definable textual portion defining the behavior of the graphical symbol.

The Final Office Action admits that Rompaey does not disclose or suggest the recited step of displaying. Appellant agrees with this characterization. However, the Final Office Action cites Hellestrand (col. 21, lines 39-59) or, alternatively, Cadence (page 2, paragraphs 1-3) as teaching the recited displaying. Appellant respectfully traverses these characterizations. Col. 21, lines 39-59 (Hellestrand) teach that prior to execution, a user can insert, enable, or disable debugger breakpoints in the user programs for each processor simulator. During simulation, when a breakpoint is encountered, the debugger stops, thereby allowing any software variable in

any processor simulator and any hardware variable in the hardware simulator to be examined by a user. Page 2, paragraphs 1-3 (Cadence) states that the Interactive Simulation Library™ (ISL), which is a library provided by Cadence, allows the control and monitoring of multiple inputs/outputs and design parameters to facilitate the testing of virtual test instruments (e.g. oscilloscopes, spectrum analyzers, and signal generators).

Appellant respectfully submits that neither reference, i.e. Hellestrand or Cadence, teaches the recited step of displaying on the GUI a graphical representation of the execution of the software application on the virtual embedded system that includes a software debugger interface to debug the loaded software and a virtual test-bench associated with the GUI and adapted to interact with the simulation, wherein the virtual test-bench is created using a test-bench builder for generating a graphical representation of at least one interactive test-bench and for selecting signals or variables associated with the FSM and to be coupled to a graphical representation of a user interface for each interactive test-bench, to emulate user input to and device output from the virtual embedded system.

Therefore, even when Rompaey, Hellestrand, and Cadence are combined (assuming these references can be combined, which is not conceded herein), the recited steps of forming a virtual embedded system and displaying are still neither disclosed nor suggested. Based on the above reasons, Appellant requests reconsideration and withdrawal of the rejection of Claim 37.

Claims 38-39 and 49-51 depend from Claim 37 and therefore are patentable for at least the reasons presented for Claim 37. Based on those reasons, Appellant requests reconsideration and withdrawal of the rejection of Claims 38-39 and 49-51.

B. Claims 40-42 are patentable under 35 U.S.C. 103(a) over Rompaey, Cadence, Hellestrand, and U.S. Publication 2004/0250083 (Schwab).

1. Rompaey, Cadence, Hellestrand: Overviews see Section A.

2. Schwab: Overview

Schwab teaches a secure identification system that uses a separate, centralized database to store data-compressed images of the individuals and to download the data-compressed images to local data terminals, on demand, at the time of a transaction. Paragraph 0016.

3. The limitations recited in Claims 40-42 are not taught by Rompaey, Cadence, Hellestrand, and Schwab.

Claims 40-42 depend from Claim 37 and therefore are patentable for at least the reasons presented for Claim 37. Schwab fails to remedy the deficiency of Rompaey, Cadence, and Hellestrand with respect to Claim 37. Specifically, Schwab teaches an interactive secure identification transaction system for storing, retrieving, and displaying text and data compressed image files and communicating those between a centralized server computer and a plurality of client data terminals located at remote sites. Paragraph [0002]. Appellant submits that Schwab teaches nothing about an FSM or FSM simulation, much less the recited steps of forming a virtual embedded system or displaying. Based on the above reasons, Appellant requests reconsideration and withdrawal of the rejection of Claims 40-42.

C. Claims 43-48 are patentable under 35 U.S.C. 103(a) over Rompaey, Cadence, Hellestrand, and U.S. Patent 6,094,654 (Van Huben).

1. Rompaey, Cadence, Hellestrand: Overviews see Section A.

2. Van Huben: Overview

In col. 7, lines 28-60, Van Huben teaches making a common model by displaying control screen sections as part of a control panel input screen, which allows interactive user activity. Sections of the control screen panel can include different fields. In one embodiment, a first field can represent the name of an anchor name field of a model (which is identical to the name of a data object which is serving as a named anchor), a second field can represent a library where the named anchor resides, a third field can represent the type of data object identified by the anchor name, a fourth field can represent user entries for the version of the named anchor, a fifth field can represent user entries for the level of the named anchor for use by a user or a third party tool for creating, modifying or deleting an aggregate collection of data objects, encompassing those used for items that are identified, tabulated, tracked, validated and invalidated, and promoted, as are bills of materials, by said data management system, and a sixth field can represent user entries for the status of the named anchor.

3. The limitations recited in Claims 40-42 are not taught by Rompaey, Cadence, Hellestrand, and Van Huben.

Claims 43-48 depend from Claim 37 and therefore are patentable for at least the reasons presented for Claim 37. Van Huben fails to remedy the deficiency of Rompaey, Cadence, and Hellestrand with respect to Claim 37. Specifically, Van Huben teaches storing, moving, retrieving and managing data in a system comprised of one or more shared public libraries interacting with one or more private libraries arranged in a client server environment. Col. 6, line 65 to col. 7, line 1. Therefore, Appellant submits that Van Huben teaches nothing about an FSM or FSM simulation, much less the recited steps of

forming a virtual embedded system or displaying. Based on the above reasons, Appellant requests reconsideration and withdrawal of the rejection of Claims 43-48.

D. Claims 52-59, 64, and 89 are patentable under 35 U.S.C. 103(a) over Rompaey and Cadence.

1. Rompaey, Cadence: Overviews see Section A.

2. The limitations recited in Claims 52-59, 64, and 89 are not taught by Rompaey and Cadence.

Claim 52 recites in part (emphasis added):

> generating a virtual hardware component that is a finite state machine (FSM) representation of at least one hardware component, said generating comprising applying a design language having at least one graphical symbol and adapted to form an FSM representation of electronic hardware, each graphical symbol of the design language having a graphical portion and a user-definable textual portion defining the behavior of the graphical symbol; [and]
>
> linking read, write, and interrupt signals of the instruction set accurate simulator of the target processor core with corresponding signals of the at least one virtual hardware component to form a virtual embedded system.

Appellant respectfully submits that the cited references fail to disclose or suggest these limitations. For example, Rompaey teaches a hardware/software co-design environment that allows specifying, simulating, and synthesizing heterogeneous hardware/software architectures from a heterogeneous specification. Col. 7, lines 31-35. This environment is based on encapsulating existing hardware and software compilers and allows for the interactive synthesis of hardware/software as well as hardware/hardware interfaces. Col. 7, lines 35-39.

The Final Office Action cites the following passages of Rompaey as teaching generating the virtual hardware component that is a FSM representation: col. 9, lines 19-22; col. 20, line 55 to col. 21, line 14; and col. 11, lines 5-6. Appellant respectfully traverses this characterization. Col. 9, lines 19-22 teach that a hybrid simulation includes both hardware implementations and computer simulations, wherein the hardware implementation can include hardware and software subsystems (the software subsystems being executed on the hardware subsystems). This hybrid simulation does not teach an FSM, much less the recited generating of the virtual hardware component that is an FSM representation of at least one hardware component. Col. 20, line 55 to col. 21, line 14 teach that the ports 75 of the software model 71 (see FIG. 7) have primitive protocols, wherein the software model contains a behavioral description that allows compiling a software host language encapsulation into machine code. This software model does not teach an FSM, much less the recited generating of the virtual hardware component that is an FSM representation of at least one hardware component. Col. 11, lines 5-6 teach FIG. 11 is a schematic representation of a pager application. FIG. 11 shows Remote Procedure Call (RPC) communication for the pager design. Col. 24, lines 50-51. In this RPC communication, blocks 91-101 correspond to processes 89 (i.e. tracking and acquisition, frame extraction, correlator and noise estimator, and user interface) shown in FIG. 10. Col. 24, lines 51-53. However, these blocks do not teach an FSM, much less the recited generating of the virtual hardware component that is an FSM representation of at least one hardware component.

Thus, Appellant respectfully submits that these cited passages of Rompaey fail to disclose or suggest the recited step of generating the virtual hardware component that is a finite

state machine (FSM) representation of at least one hardware component, the generating comprising applying a design language having at least one graphical symbol and adapted to form an FSM representation of electronic hardware, each graphical symbol of the design language having a graphical portion and a user-definable textual portion defining the behavior of the graphical symbol.

The Final Office Action cites col. 9, lines 19-22 and col. 20, line 55 to col. 21, line 14 of Rompaey as teaching linking read, write, and interrupt signals of the instruction set accurate simulator of the target processor core with corresponding signals of the at least one virtual hardware component to form a virtual embedded system. Appellant respectfully traverses this characterization. As noted above, this passage teaches that the ports 75 of the software model 71 (see FIG. 7) have primitive protocols, wherein the software model contains a behavioral description that allows compiling a software host language encapsulation into machine code. Merely providing ports does not teach linking the recited signals to form the virtual embedded system. Therefore, Appellant submits that Rompaey also fails to disclose or suggest the recited linking step.

Appellant respectfully submits that Cadence fails to remedy the deficiency of Rompaey with respect to Claim 52. Specifically, Cadence fails to disclose or suggest the recited steps of generating the virtual hardware component and linking read, write, and interrupt signals.

Therefore, even when Rompaey and Cadence are combined (assuming these references can be combined, which is not conceded herein), the recited steps of generating the virtual hardware component and linking read, write, and interrupt signals are still neither disclosed nor suggested. Based on the

above reasons, Appellant requests reconsideration and withdrawal of the rejection of Claim 52.

Claims 53-59 depend from Claim 52 and therefore are patentable for at least the reasons presented for Claim 52. Based on those reasons, Appellant requests reconsideration and withdrawal of the rejection of Claims 53-59.

Claim 64 recites in part (emphasis added):

> generating a finite state machine (FSM) representation of at least one hardware element, said generating comprising applying a design language having at least one graphical symbol and adapted to form a finite state machine representation of electronic hardware, each graphical symbol of the design language having a graphical portion and a user-definable textual portion defining the behavior of the graphical symbol;
>
> designing a virtual prototype of the embedded system having an instruction set accurate simulator of a processor core and coupling read, write, and interrupt signals of the instruction set accurate simulator with said FSM representation of at least one hardware element; [and]
>
> creating a virtual test bench having a graphical representation of a human/machine interface for interacting with the embedded system using a test bench builder for generating a graphical representation of at least one interactive test bench and for selecting signals or variables associated with the FSM and to be coupled to a graphical representation of a user interface for each interactive test bench.

Appellant respectfully submits that the cited references fail to disclose or suggest these limitations. For example, Rompaey teaches a hardware/software co-design environment that allows specifying, simulating, and synthesizing heterogeneous hardware/software architectures from a heterogeneous specification. Col. 7, lines 31-35. This environment is based on encapsulating existing hardware and software compilers and

allows for the interactive synthesis of hardware/software as well as hardware/hardware interfaces. Col. 7, lines 35-39.

The Final Office Action cites the following passages of Rompaey as teaching generating a FSM representation of at least one hardware element: col. 9, lines 19-22; col. 20, line 55 to col. 21, line 14; and col. 11, lines 5-6. Appellant respectfully traverses this characterization. Col. 9, lines 19-22 teach that a hybrid simulation includes both hardware implementations and computer simulations, wherein the hardware implementation can include hardware and software subsystems (the software subsystems being executed on the hardware subsystems). This simulation does not teach an FSM, much less the recited generating of the FSM representation of at least one hardware element. Col. 20, line 55 to col. 21, line 14 teach that the ports 75 of the software model 71 (see FIG. 7) have primitive protocols, wherein the software model contains a behavioral description that allows compiling a software host language encapsulation into machine code. This software model does not teach an FSM, much less the recited generating of the FSM representation of at least one hardware element. Col. 11, lines 5-6 teach FIG. 11 is a schematic representation of a pager application. FIG. 11 shows Remote Procedure Call (RPC) communication for the pager design. Col. 24, lines 50-51. In this RPC communication, blocks 91-101 correspond to processes 89 (i.e. tracking and acquisition, frame extraction, correlator and noise estimator, and user interface) shown in FIG. 10. Col. 24, lines 51-53. However, these blocks do not teach an FSM, much less the recited generating of the FSM representation of at least one hardware element.

Appellant respectfully submits that these cited passages of Rompaey fail to disclose or suggest the recited step of generating the FSM representation of at least one hardware

element, the generating comprising applying a design language having at least one graphical symbol and adapted to form an FSM representation of electronic hardware, each graphical symbol of the design language having a graphical portion and a user-definable textual portion defining the behavior of the graphical symbol.

The Final Office Action cites col. 20, line 55 to col. 21, line 14 of Rompaey as teaching designing the virtual prototype of the embedded system. Appellant respectfully traverses this characterization. As noted above, this passage teaches that the ports 75 of the software model 71 (see FIG. 7) have primitive protocols, wherein the software model contains a behavioral description that allows compiling a software host language encapsulation into machine code. Merely providing ports does not teach designing a virtual prototype of the embedded system having an instruction set accurate simulator of a processor core and coupling read, write, and interrupt signals of the instruction set accurate simulator with said FSM representation of at least one hardware element. Therefore, Appellant submits that Rompaey also fails to disclose or suggest the recited designing step.

The Final Office Action admits that Rompaey fails to disclose or suggest the step of creating the virtual test bench. Appellant agrees with this characterization. However, the Final Office Action then cites Cadence, page 2, paragraphs 1-3 as teaching this step. Appellant respectfully traverses this characterization. Specifically, Cadence fails to disclose or suggest a graphical representation of at least one interactive test bench and for selecting signals or variables associated with the FSM and to be coupled to a graphical representation of a user interface for each interactive test bench.

Therefore, even when Rompaey and Cadence are combined (assuming these references can be combined, which is not conceded herein), the recited steps of generating the FSM representation, designing the virtual prototype, and creating the virtual test bench are still neither disclosed nor suggested. Based on the above reasons, Appellant requests reconsideration and withdrawal of the rejection of Claim 64.

Claim 89 depends from Claim 64 and therefore is patentable for at least the reasons presented for Claim 64. Based on those reasons, Appellant requests reconsideration and withdrawal of the rejection of Claim 89.

E. Claims 60-63 are patentable under 35 U.S.C. 103(a) over Rompaey, Cadence, and Van Huben.

1. Rompaey, Cadence: Overviews see Section A.

2. Van Huben: Overview see Section C.

3. The limitations recited in Claims 60-63 are not taught by Rompaey, Cadence, and Van Huben.

Claims 60-63 depend from Claim 52 and therefore are patentable for at least the reasons presented for Claim 52. Van Huben fails to remedy the deficiency of Rompaey and Cadence with respect to Claim 52. Specifically, Van Huben teaches storing, moving, retrieving and managing data in a system comprised of one or more shared public libraries interacting with one or more private libraries arranged in a client server environment. Col. 6, line 65 to col. 7, line 1. Therefore, Appellant submits that Van Huben teaches nothing about the recited steps of generating the virtual hardware component and linking read, write, and interrupt signals. Based on the above reasons, Appellant

requests reconsideration and withdrawal of the rejection of Claims 60-63.

F. Claims 65-67 are patentable under 35 U.S.C. 103(a) over Rompaey, Cadence, and U.S. Publication 2005/0193280 (Schubert).

1. Rompaey, Cadence: Overviews see Section A.

2. Schubert: Overview

3. The limitations recited in Claims 65-67 are not taught by Rompaey, Cadence, and Schubert.

Claims 65-67 depend from Claim 64 and therefore are patentable for at least the reasons presented for Claim 64. Schubert fails to remedy the deficiency of Rompaey and Cadence with respect to Claim 64. Specifically, Schubert teaches techniques and systems for analysis, diagnosis and debugging fabricated hardware designs at a Hardware Description Language (HDL) level. Paragraph [0024]. However, Schubert teaches nothing about designing a virtual prototype or creating the virtual test bench. Based on the above reasons, Appellant requests reconsideration and withdrawal of the rejection of Claims 65-67.

G. Claims 68-70 are patentable under 35 U.S.C. 103(a) over Rompaey, Cadence, and Van Huben.

1. Rompaey, Cadence: Overviews see Section A.

2. Van Huben: Overview see Section C.

3. The limitations recited in Claims 68-70 are not taught by Rompaey, Cadence, and Van Huben.

Claim 68 recites in part (emphasis added):

> designing a virtual prototype of the embedded system, the virtual prototype having an instruction set accurate simulator of a target processor core and a finite state machine (FSM) representation of a hardware element within the embedded system, the FSM representation configured to couple memory read/write requests and interrupt signals with the instruction set accurate simulator of the target processor core, wherein generating said FSM representation comprises applying a design language having at least one graphical symbol and adapted to form a finite state machine representation of electronic hardware, each graphical symbol of the design language having a graphical portion and a user-definable textual portion defining the behavior of the graphical symbol;
>
> creating a virtual test bench having a graphical representation of a human/machine interface for interacting with the embedded system using a test bench builder for generating a graphical representation of at least one interactive test bench and for selecting signals or variables associated with the FSM and to be coupled to a graphical representation of a user interface for each interactive test bench.

Appellant respectfully submits that the cited references fail to disclose or suggest these limitations. For example, Rompaey teaches a hardware/software co-design environment that allows specifying, simulating, and synthesizing heterogeneous hardware/software architectures from a heterogeneous specification. Col. 7, lines 31-35. This environment is based on encapsulating existing hardware and software compilers and allows for the interactive synthesis of hardware/software as well as hardware/hardware interfaces. Col. 7, lines 35-39.

The Final Office Action cites FIG. 11 and the following passages of Rompaey as teaching forming the virtual prototype of the embedded system: col. 20, line 55 to col. 21, line 14; and col. 11, lines 5-6. Appellant respectfully traverses this characterization. Col. 20, line 55 to col. 21, line 14 teach that the ports 75 of the software model 71 (see FIG. 7) have

primitive protocols, wherein the software model contains a behavioral description that allows compiling a software host language encapsulation into machine code. This software model does not teach an FSM, much less the recited FSM representation configured to couple requests and interrupt signals with the instruction set accurate simulator. Col. 11, lines 5-6 teach FIG. 11 is a schematic representation of a pager application. FIG. 11 shows Remote Procedure Call (RPC) communication for the pager design. Col. 24, lines 50-51. In this RPC communication, blocks 91-101 correspond to processes 89 (i.e. tracking and acquisition, frame extraction, correlator and noise estimator, and user interface) shown in FIG. 10. Col. 24, lines 51-53. However, these blocks do not teach an FSM, much less the recited FSM representation configured to couple requests and interrupt signals with the instruction set accurate simulator.

Appellant respectfully submits that FIG. 11 and these cited passages of Rompaey fail to disclose or suggest the recited step of designing a virtual prototype of the embedded system, the virtual prototype having an instruction set accurate simulator of a target processor core and a finite state machine (FSM) representation of a hardware element within the embedded system, the FSM representation configured to couple memory read/write requests and interrupt signals with the instruction set accurate simulator of the target processor core, wherein generating said FSM representation comprises applying a design language having at least one graphical symbol and adapted to form a finite state machine representation of electronic hardware, each graphical symbol of the design language having a graphical portion and a user-definable textual portion defining the behavior of the graphical symbol.

The Final Office Action admits that Rompaey does not disclose or suggest the recited step of creating a virtual test

bench. Appellant agrees with this characterization. However, the Final Office Action then cites Cadence (page 2, paragraphs 1-3) as teaching the recited step of creating. Appellant respectfully traverses these characterizations. Page 2, paragraphs 1-3 (Cadence) states that the Interactive Simulation Library™ (ISL), which is a library provided by Cadence, allows the control and monitoring of multiple inputs/outputs and design parameters to facilitate the testing of virtual test instruments (e.g. oscilloscopes, spectrum analyzers, and signal generators).

Thus, Appellant respectfully submits that Cadence does not teach the recited step of creating a virtual test bench having a graphical representation of a human/machine interface for interacting with the embedded system using a test bench builder for generating a graphical representation of at least one interactive test bench and for selecting signals or variables associated with the FSM and to be coupled to a graphical representation of a user interface for each interactive test bench.

Van Huben fails to remedy the deficiency of Rompaey and Cadence with respect to Claim 68. Specifically, Van Huben teaches storing, moving, retrieving and managing data in a system comprised of one or more shared public libraries interacting with one or more private libraries arranged in a client server environment. Col. 6, line 65 to col. 7, line 1. Therefore, Appellant submits that Van Huben teaches nothing about the recited steps of designing a virtual prototype of the embedded system or creating a virtual test bench.

Therefore, even when Rompaey, Cadence, and Van Huben are combined (assuming these references can be combined, which is not conceded herein), the recited steps of designing a virtual prototype of the embedded system and creating a virtual test bench are still neither disclosed nor suggested. Based on the

above reasons, Appellant requests reconsideration and withdrawal of the rejection of Claim 68.

Claims 69-70 depend from Claim 68 and therefore are patentable for at least the reasons presented for Claim 68. Based on those reasons, Appellant requests reconsideration and withdrawal of the rejection of Claims 69-70.

H. Claims 71-72 are patentable under 35 U.S.C. 103(a) over Rompaey, Cadence, Van Huben, and Schwab.

1. Rompaey, Cadence: Overviews see Section A.

2. Van Huben: Overview see Section C.

3. Schwab: Overview see Section B.

4. The limitations recited in Claims 71-72 are not taught by Rompaey, Cadence, Van Huben, and Schwab.

Claims 71-72 depend from Claim 68 and therefore are patentable for at least the reasons presented for Claim 68. Schwab fails to remedy the deficiency of Rompaey, Cadence, and Van Huben with respect to Claim 68. Specifically, Schwab teaches an interactive secure identification transaction system for storing, retrieving, and displaying text and data compressed image files and communicating those between a centralized server computer and a plurality of client data terminals located at remote sites. Paragraph [0002]. Appellant submits that Schwab teaches nothing about an FSM, much less the recited steps of designing a virtual prototype of the embedded system or creating a virtual test bench. Based on the above reasons, Appellant requests reconsideration and withdrawal of the rejection of Claims 71-72.

I. Claims 74 and 76-78 are patentable under 35 U.S.C. 103(a) over Van Huben, Rompaey, and Cadence.

1. Rompaey, Cadence: Overviews see Section A.

2. Van Huben: Overview see Section C.

3. The limitations recited in Claims 74 and 76-78 are not taught by Rompaey, Cadence, and Van Huben.

Claim 74 recites in part (emphasis added):

> accessing a database of virtual prototypes of embedded systems, each of the virtual prototypes having a processor simulator, a finite state machine (FSM) representation of hardware peripherals, and a virtual test bench emulating a human/machine interface for interacting with a simulation of the operation of the virtual prototype,
>
> wherein generating said FSM representation comprises applying a design language having at least one graphical symbol and adapted to form a finite state machine representation of electronic hardware, each graphical symbol of the design language having a graphical portion and a user-definable textual portion defining the behavior of the graphical symbol, and
>
> wherein the virtual test bench is created using a test bench builder for generating a graphical representation of at least one interactive test bench and for selecting signals or variables associated with the FSM and to be coupled to a graphical representation of a user interface for the interactive test bench.

Appellant respectfully submits that the cited references fail to disclose or suggest these limitations. For example, Rompaey teaches a hardware/software co-design environment that allows specifying, simulating, and synthesizing heterogeneous hardware/software architectures from a heterogeneous specification. Col. 7, lines 31-35. This environment is based on encapsulating existing hardware and software compilers and

allows for the interactive synthesis of hardware/software as well as hardware/hardware interfaces. Col. 7, lines 35-39.

The Final Office Action cites the following passages of Rompaey as teaching accessing a database of virtual prototypes of embedded systems: col. 9, lines 19-22; col. 13, lines 39-48; col. 20, line 55 to col. 21, line 14; and col. 11, lines 5-6. Appellant respectfully traverses this characterization. Col. 9, lines 19-22 teach a heterogeneous implementation comprising hardware subsystems and software subsystems, wherein the software subsystems can be executed on one or more hardware subsystems. This passage fails to teach a virtual prototype having an FSM representation of hardware peripherals. Col. 13, lines 39-48 teach an interface synthesis whereby each communication channel is refined by selection of a communication scenario. This passage also fails to teach that each virtual prototype has an FSM representation of hardware peripherals. Col. 20, line 55 to col. 21, line 14 teach that the ports 75 of the software model 71 (see FIG. 7) have primitive protocols, wherein the software model contains a behavioral description that allows compiling a software host language encapsulation into machine code. This software model fails to teach an FSM, much less the recited FSM representation of hardware peripherals. Col. 11, lines 5-6 teach FIG. 11 is a schematic representation of a pager application. FIG. 11 shows Remote Procedure Call (RPC) communication for the pager design. Col. 24, lines 50-51. In this RPC communication, blocks 91-101 correspond to processes 89 (i.e. tracking and acquisition, frame extraction, correlator and noise estimator, and user interface) shown in FIG. 10. Col. 24, lines 51-53. However, these blocks fail to teach an FSM, much less the recited FSM representation of hardware peripherals.

Appellant respectfully submits that these cited passages of

Rompaey fail to disclose or suggest the recited step of accessing a database of virtual prototypes of embedded systems, each of the virtual prototypes having a … a finite state machine (FSM) representation of hardware peripherals, … wherein generating said FSM representation comprises applying a design language having at least one graphical symbol and adapted to form a finite state machine representation of electronic hardware, … wherein the virtual test bench is created using a test bench builder for … selecting signals or variables associated with the FSM and to be coupled to a graphical representation of a user interface for the interactive test bench.

The Final Office Action cites Cadence (page 2, paragraphs 1-3) as teaching the recited virtual test bench. Appellant respectfully traverses this characterization. Page 2, paragraphs 1-3 (Cadence) states that the Interactive Simulation Library™ (ISL), which is a library provided by Cadence, allows the control and monitoring of multiple inputs/outputs and design parameters to facilitate the testing of virtual test instruments (e.g. oscilloscopes, spectrum analyzers, and signal generators).

Thus, Appellant respectfully submits that Cadence does not teach the recited virtual test bench for generating a graphical representation of at least one interactive test bench and for selecting signals or variables associated with the FSM and to be coupled to a graphical representation of a user interface for the interactive test bench.

Van Huben fails to remedy the deficiency of Rompaey and Cadence with respect to Claim 74. Specifically, Van Huben teaches storing, moving, retrieving and managing data in a system comprised of one or more shared public libraries interacting with one or more private libraries arranged in a client server environment. Col. 6, line 65 to col. 7, line 1.

Therefore, Appellant submits that Van Huben teaches nothing about the recited virtual prototypes, each having an FSM representation of hardware peripherals.

Therefore, even when Rompaey, Cadence, and Van Huben are combined (assuming these references can be combined, which is not conceded herein), the recited accessing of the database of virtual prototypes of embedded systems is still neither disclosed nor suggested. Based on the above reasons, Appellant requests reconsideration and withdrawal of the rejection of Claim 74.

Claims 76-78 depend from Claim 74 and therefore are patentable for at least the reasons presented for Claim 74. Based on those reasons, Appellant requests reconsideration and withdrawal of the rejection of Claims 76-78.

J. Claim 75 is patentable under 35 U.S.C. 103(a) over Van Huben, Rompaey, Cadence, and Schwab.

1. Rompaey, Cadence: Overviews see Section A.

2. Van Huben: Overview see Section C.

3. Schwab: Overview see Section B.

4. The limitations recited in Claim 75 is not taught by Rompaey, Cadence, Van Huben, and Schwab.

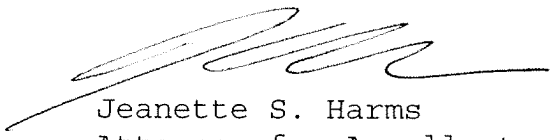
Claim 75 depends from Claim 74 and therefore is patentable for at least the reasons presented for Claim 74. Schwab fails to remedy the deficiency of Van Huben, Rompaey, and Cadence with respect to Claim 74. Specifically, Schwab teaches an interactive secure identification transaction system for storing, retrieving, and displaying text and data compressed image files and communicating those between a centralized server computer and a plurality of client data terminals located at

remote sites. Paragraph [0002]. Appellant submits that Schwab teaches nothing about an FSM representation of hardware peripherals. Based on the above reasons, Appellant requests reconsideration and withdrawal of the rejection of Claim 75.

K. CONCLUSION

For the foregoing reasons, it is submitted that the Examiner's rejections of Claims 37-72, 74-78, and 89 are erroneous, and reversal of these rejections is respectfully requested.

Respectfully submitted,



Customer No.: 35273

Jeanette S. Harms
Attorney for Appellant
Reg. No. 35,537

Telephone: 408-451-5907
Facsimile: 408-451-5908

VIII. CLAIMS APPENDIX

1. (Withdrawn) A computer system for simulating the operation of an embedded system using a software application residing in the memory of a computer, comprising:

a design application configured to form a hardware specification of an embedded system, the design application including:

a design language having at least one graphical symbol adapted to form a finite state machine (FSM) representation of electronic hardware, each graphical symbol of the design language having a graphical portion and a user-definable textual portion defining the behavior of the graphical symbol,

a library including at least one instruction set accurate simulator for simulating the behavior of a target processor core having at least one memory read pin, at least one write memory pin, and at least one interrupt pin,

a graphical user interface adapted to permit a user to select an instruction set accurate simulator of a target processor core and to form a FSM representation of a hardware component in the design language, and

a configuration interface for coupling memory read/write signals and interrupt signals of the instruction set accurate simulator to corresponding signals of a the hardware component;

a test bench builder for generating a graphical representation of at least one interactive test bench and for selecting signals or variables to be coupled to a graphical representation of a user interface for each interactive test bench;

a software debugger including a debugging interface for

associating a breakpoint of execution a graphical symbol of the finite state machine representation of the hardware component;

a graphical database and a behavioral database coupling data associated with the hardware description to a code generator;

a compiler for compiling the output of the code generator into an object file of the hardware description;

a discrete event simulation engine configured to execute the object file of the hardware description;

a first API interface adapted to couple the discrete event simulation engine to each of the interactive test benches;

at least one API interface adapted to couple the discrete event simulation engine to the instruction set accurate simulator of the hardware description; and

at least one API interface adapted to couple the instruction set accurate simulator to a software debugger, the software debugger configured to load at least one binary program code compiled for the target processor.

2. (Withdrawn) The computer system of claim 1, wherein the debugging interface is configured to permit the at least one compiled binary program code to be stepped to the break points of execution.

3. (Withdrawn) The computer system of claim 1, wherein the design language is an adaptation of the specification and description language (SDL).

4. (Withdrawn) The computer system of claim 1, wherein the computer system has a simulation speed selected so that the software application executes at a rate greater than one million instructions per second.

5. (Withdrawn) The computer system of claim 1, wherein the computer system serves as a hardware emulator and has a simulation speed sufficient to boot an embedded system operating system is less than one minute.

6. (Withdrawn) The computer system of claim 1, wherein the computer system is coupled to a server of a computer network.

7. (Withdrawn) The computer system of claim 6, wherein the computer network is selected from the group consisting of: a wide area network, a local area network, an Intranet, an extranet, or a computer network coupled to the Internet.

8. (Withdrawn) The computer system of claim 6, wherein a client computer may be coupled to the computer system by a network connection.

9. (Withdrawn) The computer system of claim 6, further comprising: a design repository coupled to the server adapted to store a design specification for each of a plurality of embedded systems, each design specification including sufficient information to generate the hardware specification for the embedded system.

10. (Withdrawn) The computer system of claim 9, further comprising a matching engine coupled to the design repository for associating metadata with each design specification and matching vendors to the stored design specifications based on the metadata.

11. (Withdrawn) The computer system of claim 9, wherein

each design specification includes at least one design specification source file configured to permit the hardware specification to be viewed in the design language and at least one executable file to run a simulation of the embedded system that includes at least one of the interactive test benches.

12. (Withdrawn) The computer system of claim 9, wherein each design specification includes at least one dynamic link library (DLL) that may be loaded by a loader executable to permit the design specification to be viewed in the design language and to run a simulation of the embedded system that includes at least one of the interactive test benches.

13. (Withdrawn) The computer system of claim 9, further comprising: a design manager configured to control access to the design specifications stored in the design repository.

14. (Withdrawn) The computer system of claim 6, wherein each design specification includes a design specification source file to describe each hardware element system, a file for each instruction set simulator of the embedded system, and a dynamic link library to execute the design specification.

15. (Withdrawn) The computer system of claim 9, wherein the server is coupled to the client computer via the Internet.

16. (Withdrawn) The computer system of claim 15, further comprising: an on-line bulletin board coupled to the server for a user to post a design specification.

17. (Withdrawn) The computer system of claim 16, wherein the bulletin board is a bidding board.

18. (Withdrawn) The computer system of claim 9, further comprising: content and a content viewer configured to interactively explain to a user at least one attribute of a selected embedded system.

19. (Withdrawn) The computer system of claim 1, further comprising: a walkthrough application including content and a content viewer configured to interactively explain to a user at least one attribute of an executable file of the hardware description of a selected embedded system.

20. (Withdrawn) The computer system of claims 18 or 19, wherein the walkthrough application is configured to allow the user to control the operation of a simulation of the embedded system through the content viewer.

21. (Withdrawn) The computer system of claim 20, wherein the embedded system includes a processor core and at least one reference hardware peripheral for evaluating the processor core.

22. (Withdrawn) A computer-implemented method for assisting a user to evaluate at least one component of an embedded system, the method comprising:

forming a virtual embedded system including an instruction set accurate simulator of a target processor core coupling read, write, and interrupt signals with a finite state machine (FSM) simulation of at least one hardware element;

coupling a virtual test bench emulating a human/machine interface to the virtual embedded system, the virtual test bench having a graphical user interface adapted to interact with the virtual embedded system;

receiving from the user a binary program code of a software application compiled for the target processor core; and

responsive to a request from a user, loading the software application and running a simulation of the embedded system.

23. (Withdrawn) The method of claim 22, further comprising:

creating content for explaining the operation of the virtual embedded system; and

responsive to a user request, providing the content as an interactive demonstration.

24. (Withdrawn) The method of claim 22, wherein the virtual embedded system resides on a server and the method further comprises:

forming a data connection between the server and a client computer;

receiving command and data user inputs from a web browser residing on the client computer; and

providing graphical display data of the simulation running on the server to the web browser residing on the client computer.

25. (Withdrawn) The method of claim 24, wherein the server is a network server coupled to the client computer via the Internet and the method further comprises:

monitoring the user's interaction with the virtual embedded system; and

recording data associated with an interaction of the user with the virtual embedded system.

26. (Withdrawn) The method of claim 24, further comprising:

forming a plurality of virtual embedded systems in a

library;

receiving a request from a user for a selected virtual embedded system; and

recording data associated with usage of the selected virtual embedded system.

27. (Withdrawn) The method of claim 26, further comprising:

requesting the user to submit registration data.

28. (Withdrawn) The method of claim 25, further comprising:

providing each user with a graphical user interface to modify the virtual embedded system;

monitoring modifications made by a plurality of users to the virtual embedded system; and

determining a statistically popular modifications to the virtual embedded system.

29. (Withdrawn) The method of claim 24, wherein the software debugger is adapted to load binary program code compiled for a target processor core and the method further comprises:

responsive to a user request, loading onto the virtual embedded system a binary executable program file of a software application compiled for the processor core; and

displaying a simulation of the virtual embedded system executing the software application.

30. (Withdrawn) The method of claim 22, further comprising:

providing the user a graphical interface having a design language including a plurality of graphical symbols adapted to form a finite state representation of a user-defined hardware element that may be coupled to the virtual embedded system.

31. (Withdrawn) The method of claim 30, wherein the graphical symbols have a graphical portion and a textual portion with at least one of the graphical symbols including a textual portion adapted for the user to describe a behavior of the symbol.

32. (Withdrawn) The method of claim 31, wherein a user may input a text portion including at least one command written in the ANSI C/C++ language.

33. (Withdrawn) The method of claim 31, wherein the design language is an adaptation of a specification and description language (SDL).

34. (Withdrawn) The method of claim 22, wherein a plurality of virtual embedded systems are stored in a database, the method further comprising:

responsive to a user request, selecting one of the virtual embedded systems for evaluation.

35. (Withdrawn) The method of claim 22 further comprising:

forming content for the embedded system, the content including a at least one graphical control interface configured to interact with the simulation of the embedded system:

responsive to a user request displaying content associated with the embedded system; and

responsive to a user input to the graphical control interface, displaying an interaction with the simulation.

36. (Withdrawn) The method of claim 35, wherein the content

is a tutorial of the embedded system.

37. (Previously Presented) In a computer system having a graphical interface (GUI) and a design language for forming a finite state machine (FSM) representation of a hardware partition of an embedded system, a method of designing an embedded system, the method comprising:

forming a library of processor cores including an instruction set accurate simulator for each of the processor cores in the library;

responsive to a first sequence of user commands, selecting at least one of the processor cores from the library as a target processor core;

responsive to a second sequence of user commands, forming a virtual embedded system including an instruction set accurate simulator of a target processor core and coupling read, write, and interrupt signals of the instruction set accurate simulator with an FSM simulation of at least one hardware element, wherein generating said FSM simulation comprises applying a design language having at least one graphical symbol and adapted to form a finite state machine representation of electronic hardware, each graphical symbol of the design language having a graphical portion and a user-definable textual portion defining the behavior of the graphical symbol;

responsive to a request from the user, loading an executable binary file of a software application compiled for the target processor;

executing a simulation of the virtual embedded system running the software application; and

responsive to a user request, displaying on the GUI a graphical representation of the execution of the software application on the virtual embedded system that includes a

software debugger interface to debug the loaded software and a virtual test-bench associated with the GUI and adapted to interact with the simulation, wherein the virtual test-bench is created using a test-bench builder for generating a graphical representation of at least one interactive test-bench and for selecting signals or variables associated with the FSM and to be coupled to a graphical representation of a user interface for each interactive test-bench, to emulate user input to and device output from the virtual embedded system.

38. (Original) The method of claim 37, wherein the design language includes a plurality of graphical symbols with each graphical symbol having a graphical semantic portion and a textual semantic portion.

39. (Original) The method of claim 38, wherein the design language includes:

a start object defining a starting point of the finite state machine at an initialization time, the start object having an output connector activated when the start object is initialized;

a task object including a field for inputting computer code in the C language for defining a behavior of the task object and a connector port for coupling the task object to other objects;

a state object for representing a state of the finite state machine;

a decision object having an evaluation field for directing a flow of execution based on a result of an expression in the decision field;

a signal-out object for sending a communication signal;

a signal-in object for receiving a communication signal;

a connector object for connecting control flow;

a symbol object having at least one user-definable pin connector and containing a block or process object;

a block object for describing the behavior of one or more processes; and

a process object for representing a finite state machine process.

40. (Original) The method of claim 37, further comprising:

storing the virtual embedded system as a design in a design repository coupled to a server; and

providing access privileges to the design to a selected individual or group.

41. (Original) The method of claim 40, further comprising:

responsive to a user command, providing access privileges to the design to a group of vendors.

42. (Original) The method of claim 41, wherein the design is accessible from an online bidding board.

43. (Previously Presented) The method of claim 41, wherein access privileges to the design are provided to a group of vendors selected by the user.

44. (Original) The method of claim 40, wherein the design is accessible by an embedded system design team.

45. (Original) The method of claim 40, further comprising a design manager application, the method further comprising:

permitting one or more members of a design team to select edit privileges of at least one version of the design.

46. (Previously Presented) The method of claim 45, further comprising:

permitting one or more members of the design team to load onto the design and execute a binary program executable of a software application compiled for the target processor core.

47. (Original) The method of claim 45, further comprising:

providing version control and regulating editing access to maintain a consistent version of the design.

48. (Original) The method of claim 37, further comprising:

storing a plurality of virtual embedded systems in a library; and

responsive to a user request, providing the user a copy of one of the virtual embedded systems stored in the library.

49. (Previously Presented) The method of claim 38, wherein the graphical user interface is configured to permit a user to associate a breakpoint of execution with a graphical symbol, the method further comprising:

responsive to a user input, associating a breakpoint of execution with a graphical symbol;

receiving a request to debug software; and

stopping the simulation responsive to a command flow of the FSM representation of the hardware element reaching the graphical symbol of the breakpoint of execution.

50. (Original) The method of claim 49, further comprising:

responsive to a user request, single-stepping the simulation to sequential breakpoints of execution in the command flow of the FSM representation of hardware elements.

51. (Original) The method of claim 49, further comprising:

responsive to a user request, single-stepping the simulation by a pre-selected number of time units.

52. (Previously Presented) A computer implemented method of embedded system design, the method comprising:

selecting an instruction set accurate simulator of a target processor core;

generating a virtual hardware component that is a finite state machine (FSM) representation of at least one hardware component, said generating comprising applying a design language having at least one graphical symbol and adapted to form an FSM representation of electronic hardware, each graphical symbol of the design language having a graphical portion and a user-definable textual portion defining the behavior of the graphical symbol;

linking read, write, and interrupt signals of the instruction set accurate simulator of the target processor core with corresponding signals of the at least one virtual hardware component to form a virtual embedded system;

creating a virtual test bench using a test bench builder for generating a graphical representation of at least one interactive test bench and for selecting signals or variables to be coupled to a graphical representation of a user interface for each interactive test bench;

coupling the virtual test bench to at least one signal or variable of the virtual embedded system to simulate a human/machine interface; and

coupling a software debugger to the virtual embedded system that is configured to load and run on the virtual embedded system at least one binary program executable of a software application compiled for the target processor core.

53. (Previously Presented) The method of claim 52, further comprising:

selecting the target processor core simulator from a library having a plurality of instruction set accurate simulators for a plurality of processor cores.

54. (Original) The method of claim 52, further comprising:

selecting the virtual hardware component from a library of virtual hardware components.

55. (Original) The method of claim 54, further comprising:

modifying the virtual hardware component.

56. (Previously Presented) The method of claim 52, further comprising:

loading onto the software debugger benchmark software in an evaluation phase of an embedded system project and running a simulation of the virtual embedded system executing the benchmark software.

57. (Previously Presented) The method of claim 52, further comprising:

loading binary program executables of development software compiled for the target processor core in a development phase of an embedded systems project and running a simulation of the virtual embedded system executing the binary program executables.

58. (Original) The method of claim 57, further comprising:

debugging the development software using a software debugger.

59. (Previously Presented) The method of claim 52, further comprising:

storing the virtual embedded system in a design repository as a design having at least one executable file.

60. (Original) The method of claim 59, wherein the design is stored on a server accessible to a user-group.

61. (Original) The method of claim 60, wherein the user-group is a geographically distributed embedded system project team.

62. (Original) The method of claim 59, further comprising:

providing a version of the design to a vendor offering a good or service related to the virtual embedded system.

63. (Original) The method of claim 62, wherein the design is stored on a server and a vendor is provided access to the design via a network connection.

64. (Previously Presented) A method of designing an embedded system, the method comprising:

defining a system architecture of the embedded system;

generating a finite state machine (FSM) representation of at least one hardware element, said generating comprising applying a design language having at least one graphical symbol and adapted to form a finite state machine representation of electronic hardware, each graphical symbol of the design language having a graphical portion and a user-definable textual portion defining the behavior of the graphical symbol;

designing a virtual prototype of the embedded system having

an instruction set accurate simulator of a processor core and coupling read, write, and interrupt signals of the instruction set accurate simulator with said FSM representation of at least one hardware element;

creating a virtual test bench having a graphical representation of a human/machine interface for interacting with the embedded system using a test bench builder for generating a graphical representation of at least one interactive test bench and for selecting signals or variables associated with the FSM and to be coupled to a graphical representation of a user interface for each interactive test bench;

coupling the virtual prototype to a software debugger having a debugging interface and to the virtual test bench;

developing at least one software application for the processor core;

loading binary program code compiled from the at least one software application for execution on the virtual prototype; and

initiating a simulation of the virtual prototype executing the at least one software application.

65. (Original) The method of claim 64, further comprising:

developing a hardware implementation using the virtual prototype as a functional specification describing a hardware partition.

66. (Original) The method of claim 65, further comprising:

evaluating the operation of the embedded system executing the at least one software application; and

responsive to a result of the evaluation, modifying a hardware or software component of the embedded system.

67. (Previously Presented) The method of claim 64, wherein

the step of defining the system architecture comprises:

selecting at least one embedded system component for evaluation;

forming a virtual evaluation platform including the selected embedded system component;

loading a benchmark software application for execution on the virtual evaluation platform; and

evaluating performance of the virtual evaluation platform executing the benchmark software application.

68. (Previously Presented) A method of providing information to potential suppliers for procuring a good or service associated with an embedded system, the method comprising:

defining a system architecture of the embedded system;

designing a virtual prototype of the embedded system, the virtual prototype having an instruction set accurate simulator of a target processor core and a finite state machine (FSM) representation of a hardware element within the embedded system, the FSM representation configured to couple memory read/write requests and interrupt signals with the instruction set accurate simulator of the target processor core, wherein generating said FSM representation comprises applying a design language having at least one graphical symbol and adapted to form a finite state machine representation of electronic hardware, each graphical symbol of the design language having a graphical portion and a user-definable textual portion defining the behavior of the graphical symbol;

creating a virtual test bench having a graphical representation of a human/machine interface for interacting with the embedded system using a test bench builder for generating a graphical representation of at least one interactive test bench

and for selecting signals or variables associated with the FSM and to be coupled to a graphical representation of a user interface for each interactive test bench;

coupling the virtual prototype to the virtual test bench for emulating user interaction with the embedded system;

publishing the virtual prototype as a functional specification from which a vendor may initiate a simulation of the operation of the embedded system.

69. (Original) The method of claim 68, wherein the virtual prototype is stored on a computer readable medium and publishing the virtual prototype comprises sending the computer readable medium to a vendor.

70. (Previously Presented) The method of claim 68, wherein the virtual prototype is published by posting the virtual prototype as a design hosted on a database coupled to a network server.

71. (Original) The method of claim 70, wherein the virtual prototype is published to a bulletin board of the database.

72. (Original) The method of claim 71, wherein the bulletin board is a bidding board.

73. (Original) The method of claim 70, wherein the virtual prototype is published to a database having a matching engine.

74. (Previously Presented) A computer-implemented method for a vendor to acquire information for the procurement of a good or service related to an embedded system project, the method comprising:

accessing a database of virtual prototypes of embedded systems, each of the virtual prototypes having a processor simulator, a finite state machine (FSM) representation of hardware peripherals, and a virtual test bench emulating a human/machine interface for interacting with a simulation of the operation of the virtual prototype,

wherein generating said FSM representation comprises applying a design language having at least one graphical symbol and adapted to form a finite state machine representation of electronic hardware, each graphical symbol of the design language having a graphical portion and a user-definable textual portion defining the behavior of the graphical symbol, and

wherein the virtual test bench is created using a test bench builder for generating a graphical representation of at least one interactive test bench and for selecting signals or variables associated with the FSM and to be coupled to a graphical representation of a user interface for the interactive test bench;

instantiating an instance of one of the virtual prototypes; and

evaluating the virtual prototype.

75. (Original) The method of claim 74, further comprising:

submitting a quote for a good or service related to the embedded system simulated by the virtual prototype.

76. (Original) The method of claim 74, wherein the virtual prototype is configured to show a parts list.

77. (Original) The method of claim 74, wherein the virtual prototype is configured to show a component net list.

78. (Original) The method of claim 74, wherein the database of virtual prototypes is hosted on a network server accessible by a client computer.

79. (Withdrawn) A computer-implemented method for evaluating at least one component of an embedded system, the method comprising:

forming a virtual embedded system including an instruction set accurate simulator of a target processor core coupling read, write, and interrupt signals with a finite state machine (FSM) representation of a hardware element;

coupling a virtual test bench emulating a human/machine interface to the virtual embedded system, the virtual test bench having a graphical user interface adapted to interact with the virtual embedded system;

hosting the virtual embedded system on a network server accessible from a client computer via the Internet;

uploading from the client computer a binary executable program file of a software application compiled for the target processor core; and

responsive to a request from a user, running a simulation of the operation of the virtual embedded system executing the uploaded software application.

80. (Withdrawn) The method of claim 79, further comprising:

forming content for explaining the operation of the embedded system; and

responsive to a request from a user, sending the content to the client computer.

81. (Withdrawn) The method of claim 79, further comprising:

responsive to a request from the client computer, forming a persistent data connection between the network server and a client computer;

receiving command and data user inputs from a web browser residing on the client computer; and

providing graphical display data of the simulation running on the server to the web browser residing on the client computer.

82. (Withdrawn) The method of claim 79, further comprising:

monitoring the user's interaction with the virtual embedded system; and

recording as marketing data at least one interaction of the user with the virtual embedded system.

83. (Withdrawn) The method of claim 79, further comprising:

forming a plurality of virtual embedded systems in a library;

receiving a request from a user for a selected virtual embedded system; and

recording data associated with usage of the selected virtual embedded system.

84. (Withdrawn) The method of claim 83, further comprising:

requesting the user to submit registration data and associating the registration data with the marketing data.

85. (Withdrawn) The method of claim 84, further comprising:

providing the user a graphical interface having a design language including a plurality of graphical symbols adapted to form a finite state representation of a user-defined hardware element that may be coupled to the virtual embedded system;

monitoring modifications made by a plurality of users to the virtual embedded system; and

determining a popular modification to the virtual embedded system.

86. (Withdrawn) A computer program product for simulating an embedded system, the computer program product comprising:

a computer readable medium:

a processor simulator module stored on the medium having an instruction set accurate simulator for at least one processor core;

a design application module stored on the medium for designing a virtual prototype of the embedded system having an instruction set accurate simulator of at least one processor core coupling read, write, and interrupt signals with a finite state machine (FSM) representation of at least one hardware element;

a simulation engine module for executing a simulation of the virtual prototype;

a software debugger module stored on the medium for a user to load compiled program code for execution on the virtual prototype; and

a test bench module stored on the medium for forming a virtual test bench having a graphical representation of a human/machine interface for interacting with an embedded system prototype.

87. (Withdrawn) A computer program product for providing information on the operation of an embedded system, comprising:

a computer readable medium;

a hardware description of the virtual embedded system stored on the medium, the hardware description including an

instruction set accurate simulator of a processor core coupling read, write, and interrupt signals with a finite state machine (FSM) representation of at least one hardware element;

a simulation engine module stored on the computer readable medium for forming a simulation of the embedded system running a software application on the hardware description; and

a test bench module stored on the medium for forming a virtual test bench emulating a human/machine interface to the simulation of the embedded system, the virtual test bench having a graphical user interface adapted to interact with the simulation of the embedded system.

88. (Withdrawn) A computer program product for evaluating an embedded system, comprising:

a computer readable medium;

at least one executable file of a hardware description of a virtual embedded system stored on the medium, the virtual embedded system including an instruction set accurate simulator of a processor core coupling read, write, and interrupt signals with a finite state machine (FSM) simulation of at least one hardware element;

a simulation engine module for running the at least one executable file as a simulation of the embedded system; and

a test bench module stored on the medium configured to form a virtual test bench emulating a human/machine interface to the virtual embedded system, the virtual test bench having a graphical user interface adapted to interact with the virtual embedded system

89. (Previously Presented) The method of claim 64, further comprising the step of interacting at runtime with the virtual prototype to simulate an application of the embedded system.

IX. EVIDENCE APPENDIX

None

X. RELATED PROCEEDINGS APPENDIX

None